A Note on the Performance Analysis of Static Locking in Distributed Database Systems

Yinghong Kuang and Ravi Mukkamala
Department of Computer Science
Old Dominion University
Norfolk, Virginia 23529.

Abstract

Even though transaction deadlocks can severely affect the performance of distributed database systems, many current evaluation techniques ignore this aspect. In [4], Shyu and Li proposed an evaluation method which takes deadlocks into consideration. However, their technique is limited to exclusive locking. In this paper, we extend their technique to allow for both shared and exclusive locking. Using this technique, we illustrate the impact of deadlocks, in the presence of shared locking, on distributed database performance.

Index Terms: Distributed databases, exclusive locking, performance modeling, shared locking, static locking, two-phase locking.

1 Introduction

A distributed database system (DDS) is a collection of cooperating nodes each containing a set of data objects. A user transaction can enter such a system at any of these nodes. The receiving node, often referred to as the *coordinating* node, undertakes the task of locating the nodes that contain the data objects required by a transaction.

In order to maintain database consistency and correctness in the presence of concurrent transactions, several concurrency control protocols have been proposed [1]. Of these, locking protocols have been widely used in both commercial and research environments. In static locking, prior to start of execution, a transaction needs to acquire either a shared-lock (for read operations) or an exclusive lock (for update operations) on each of the relevant data objects. When transactions with conflicting lock requests are initiated concurrently, they could be possibly blocked due to a deadlock. Deadlocks are known to deteriorate performance in both centralized and distributed database systems [4,6]. In spite of this, several performance studies have ignored the deadlock problem in their analyses [2,5].

In [4], Shyu and Li proposed an elegant technique to evaluate the response time and throughput of transactions in a non-replicated DDS. (In the rest of the paper, we refer to this as the S-L technique.) Assuming *exclusive* locking (i.e., only write operations), they model the queue of lock requests at an object as a M/M/1 queue [3]. This results in a closed-form for the waiting time distribution at a node, expressed in terms of the average rates of arrivals of requests and the average lock-holding time. This technique consists of two stages. In the first stage, the average transaction response time and throughput are calculated by ignoring the deadlock. This is an iterative step that uses the known properties of the M/M/1 queue [3]. In the second stage, the probabilities of transaction conflicts and deadlocks are computed. These probabilities are used, in turn, to compute the response time and throughput in the presence of deadlocks.

In general, a database transaction reads from a set of data objects (the read-set) and writes on to a set of data objects (the write-set). Assuming that all accesses are write-only (as in S-L) results in the worst-case performance (with respect to deadlocks and response time) of a DDS. In this paper, we propose to extend the S-L technique to consider both the the read and the write operations of database transactions. Using the extended S-L, we evaluate the effect of deadlocks on distributed database systems.

2 Model

Except for the inclusion of read operations, our model is the same as in S-L. For the sake of completeness, we summarize the DDS model here.

- There are N nodes and D data objects (or data granules in S-L) in a DDS. The D data objects are uniformly distributed across the N nodes. A data object may be located at exactly one node.
- Each transaction accesses K data objects. Among these, $r \cdot K$ are for read-only purpose, and the rest are for read-write. (Obviously, $0 \leq r \leq 1$.) In other words, a transaction must procure $r \cdot K$ shared locks and $(1 - r) \cdot K$ exclusive locks.
- Each data object is equally likely to be accessed by a transaction.
- Transaction arrivals into the system is a Poisson process with rate λ.
- The communication delay between nodes is exponentially distributed with mean $\bar{t}$.
- The average execution time of a transaction, once the locks are obtained, is $\bar{S}$.

3 Evaluation Procedure

Since we are only proposing extensions to the S-L model, we do not intend to repeat the description of their procedure. Instead, we will discuss only the salient features of their procedure that are relevant to describe the proposed extensions.

In Stage 1 of the S-L technique, an iterative procedure is used to evaluate the response time and throughput of a DDS ignoring the possibility of deadlocks. In each iteration, the average waiting time (for exclusive lock requests) at each of the data objects is computed using estimates of the average lock-holding times from the previous iteration. By definition, no two exclusive lock requests can have lock grants on the same object simultaneously. Also, assuming that the lock-holding time is exponentially distributed (with mean $1/\mu$) and that the lock request arrivals form a Poisson process (with rate $\lambda_r = \lambda \cdot K/D$), the distribution of waiting time W_i at an object i is expressed as (M/M/1 queueing formula [3])

$$f_{W_i}(y) = (1 - \rho) \cdot \mu_0(y) + \lambda_r(1 - \rho) \cdot e^{-\mu(1-\rho)y} \quad (1)$$

where $\mu_0(\cdot)$ is the impulse function and $\rho = \lambda_r/\mu$. Using the waiting time distribution, the waiting times at the K data objects are randomly generated. These are used, in turn, to derive new estimates for the lock-holding times $(1/\mu)$. The iterations stop when two successive computations of average waiting time estimates are very close.

When we consider both shared and exclusive locks, the problem of estimating the waiting time distributions becomes difficult. Since two shared lock grants on the same object may exist simultaneously, and an exclusive lock may not be granted while another shared or exclusive lock is already granted, the queueing discipline at a node is complex. Such complex queueing disciplines are analytically intractable [3]. For this reason, we propose to use simulation to solve the queueing model. Given the total rate of arrival of lock requests λ_r, the shared lock ratio (r), and the average lock-holding time $(1/\mu)$, the queue at an object may be simulated. From here, the waiting time distribution may be obtained in the form of a table. Once the waiting time distribution is obtained, the same iterative procedure as in Stage 1 of S-L may be adopted to compute the response time when deadlocks are ignored. As in S-L, transaction response time is defined as the time between the instance the lock requests are sent and the time the last grant request is received by the coordinating node.

In Stage 2, the probabilities of transaction deadlock and restart are computed. These are then used to compute response time and throughput in the presence of deadlocks. When we assume that transactions only make exclusive lock requests, the expression for the probability of conflict between any two transactions is given by,

$$P_c = 1 - \frac{\binom{D-K}{K}}{\binom{D}{K}} \tag{2}$$

However, when we consider both shared locks and exclusive locks, the probability of conflict is reduced. In this case the probability of conflict is given by,

$$P_c' = 1 - \frac{\binom{D-K}{K}}{\binom{D}{K}} - \frac{\sum_{i=1}^{K'} \binom{K'}{i} \cdot \binom{D-K}{K'-i} \cdot \binom{D-K-K'+i}{K-K'}}{\binom{D}{K} \cdot \binom{K}{K'}} \tag{3}$$

where $K' = r \cdot K$ and represents the average number of shared locks; $(K - K')$ is the average number of exclusive locks per transaction. Clearly, when $r = 0$, $P_c = P_c'$; when $r = 1$, $P_c' = 0$; and in all cases, $P_c \geq P_c'$.

By replacing P_c with P_c', the procedure suggested in S-L may be applied to obtain the desired performance metrics.

4 Results

Using the extended S-L technique, we obtained a number of interesting results that illustrate the effect of deadlocks on database performance. These are summarized in Figures 1-5. We have verified our results with those obtained in [4] for the all exclusive locks case ($r = 0$). We make the following observations.

- As expected, the presence of shared locks has a substantial impact on the probability of deadlock occurrence (Fig. 1). When only 1/3 of the accessed data objects are updated (i.e., $r = 2/3$), the probability of deadlock is considerably small as compared to when all objects are updated ($r = 0$).

- The observations about the deadlock probabilities are also valid for restart probabilities (Fig. 2).

- Transaction response times are also quite sensitive to the ratio of shared locks (Fig. 3). Here, we compare the response times when deadlocks are ignored (computed in Stage 1) with those obtained when deadlocks are considered (computed in Stage 2). The effect of deadlocks is more predominant at higher transaction loads and with smaller values of r. When $r = 2/3$, the effect of deadlocks is not significant on response time.

- The effect of deadlocks on response time is decreased with the increase in the number of data items (Fig. 4). Obviously, this is due to the decrease in probability of conflicts and hence a decrease in deadlock occurrence. For $r = 2/3$, this effect is almost insignificant. For $r = 1/3$ and $r = 0$, deadlocks seems to have a noticeable effect on response time.

- Fig. 5 summarizes the effect of the number of locks per transaction on response time. When K is small, the probability of deadlock is negligible, and hence its effect on response time is small. At higher values of K, the effect of deadlocks on response times is significant. Similarly, at smaller values of r, the effect of dedalocks is more apparent.

5 Conclusion

In [4], Shyu and Li presented an elegant technique to evaluate the performance of distributed database systems in the presence of deadlocks. Their technique assumed only exclusive locks and thus representing the worst-case effects of deadlocks. In this paper, we have extended their technique to allow both shared and exclusive locking. Using the extended technique, we evaluated the the effect of number of data objects, the number of data objects accessed, and the ratio of read operations on transaction response time. These results also indicate the importance of considering both shared and exclusive lock requests for response time evaluations.

References

[1] P. A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, 1987.

[2] A. Hac, "A decomposition solution to a queueing network model of a distributed file system with dynamic locking," *IEEE Trans. Software Eng.*, vol. SE-12, no. 4, pp. 521-530, Apr. 1986.

[3] L. Kleinrock, *Queueing Systems, Vol. I*, New York: Wiley-Interscience, 1975.

[4] S.-C. Shyu and V. O. K. Li, "Performance analysis of static locking in distributed database systems," *IEEE Trans. Computers*, vol. 39, no. 6, pp. 741-751, June 1990.

[5] M. Singhal and A. K. Agrawala, "Performance analysis of an algorithm for concurrency control in replicated database systems," *Proc. ACM SIGMETRICS Conf. Measurement Modeling Comput. Syst.*, 1986, pp. 159-169.

[6] Y. C. Tay, R. Suri, and N. Goodman, "A mean value performance model for locking in databases: The no-waiting case," *J. ACM*, vol. 32, no. 3, pp. 618-651, July 1985.

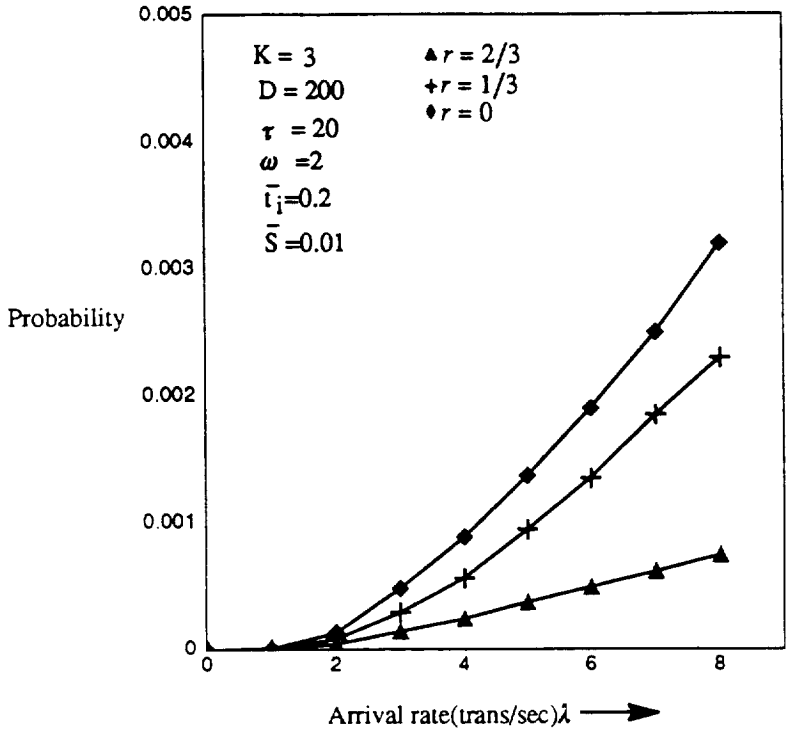

Fig.1. Deadlock probability with different read ratios

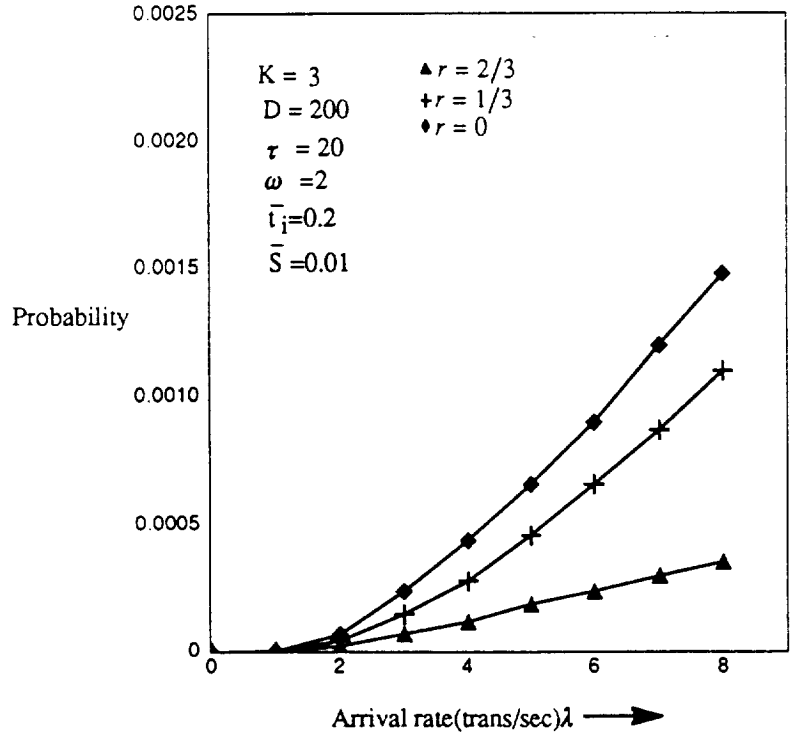

Fig.2. Restart probability with different read ratios

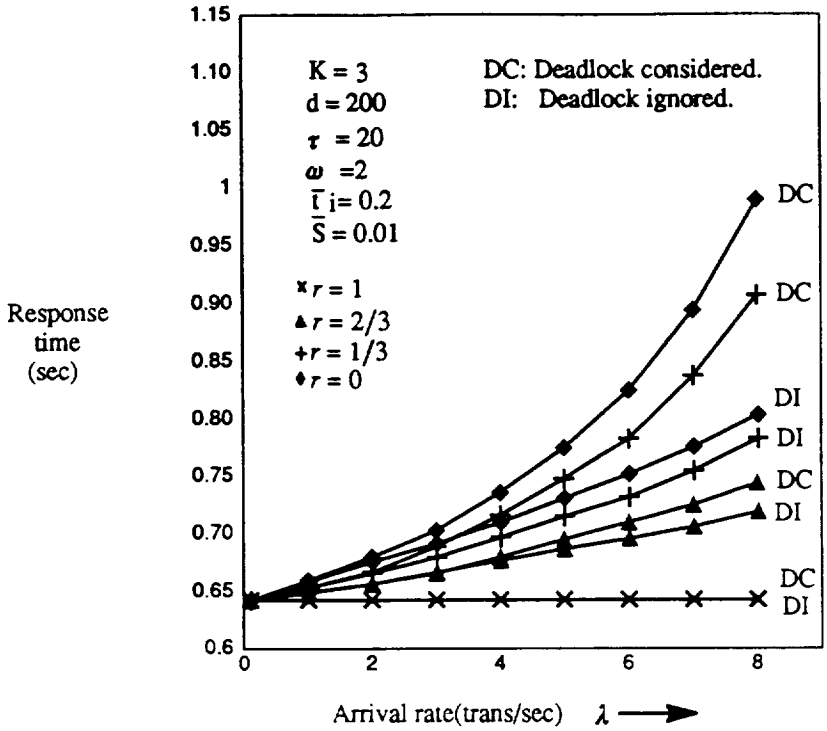

Fig.3 Comparison of response time when deadlock is considered and deadlock is ignored.

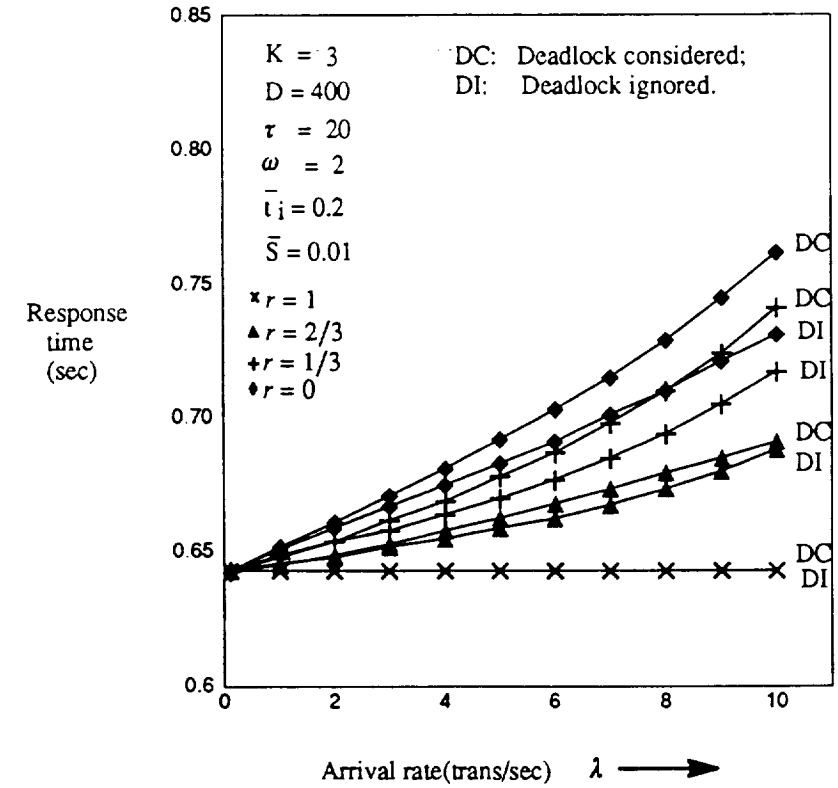

Fig. 4 Response time with high number of data objects.

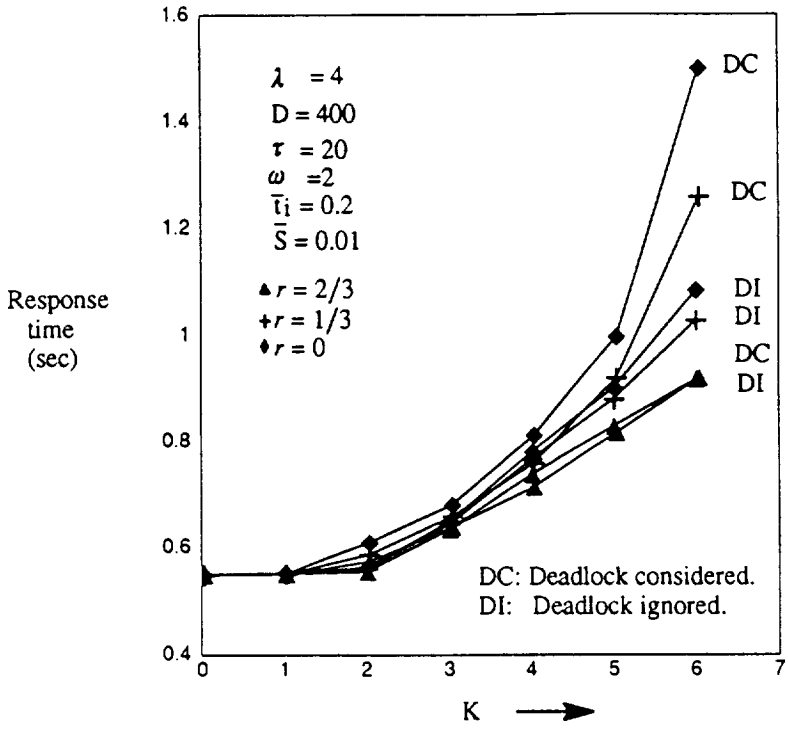

Fig.5. Comparison of response time with different number of lock requests.